平成 21 年度 (3 年生後期)

実験実習

Experiments in Electric Engineering

—ライントレースカーの製作実習—

秋田工業高等専門学校　電気情報工学科

作成日　　　年　　月　　日

作成者

目 次

実験テーマ 307　ライントレースカーの製作実習 1

1 目的 1

2 実験のスケジュール・場所 1

3 原理 1

4 実験機材と部品の使用上の注意 2
4.1 実験機材 . . . 2
4.2 使用上の注意 . . . 3
4.2.1 半導体 . . . 3
4.2.2 電池 . . . 5
4.2.3 ブレッドボード . . . 5

5 実験方法 6
5.1 車体の製作 . . . 6
5.2 回路作成 . . . 7
5.2.1 ブレッドボー上の H8 のピン配置 . . . 7
5.2.2 制御回路 . . . 8
5.3 ハードウェアーの確認 . . . 10
5.4 プログラム作成 . . . 11
5.4.1 ソフトウェアーについて . . . 11
5.4.2 制御プログラムの作成と転送 . . . 11
5.5 実行 . . . 13
5.6 改良 . . . 13
5.6.1 ハードウェアーの改良 . . . 13
5.6.2 ソフトウェアーの改良 . . . 14
5.7 秋田高専グランプリ . . . 15

6 考察課題 16

7 レポート作成上の注意 16
7.1 書き方の注意 . . . 16
7.2 記述内容 . . . 17
7.3 提出要領 . . . 18

A ライントレースカー構成部品の動作 20
A.1 H8 マイコン . . . 20
A.2 赤外線センサー . . . 20
A.3 DC モーターと PWM 制御 . . . 21
A.3.1 DC モーターの構造 . . . 21
A.3.2 DC モーターの PWM 制御 . . . 22
A.4 トランジスター . . . 23
A.5 ダイオード . . . 24

B ライントレースカーのソフトウェアー 26
B.1 メイクファイル 26
B.2 ライントレースカー用の関数 27
B.2.1 ヘッダーファイル 27
B.2.2 制御用の関数 27
B.3 H8 マイコン 3664 のヘッダーファイル 29
B.4 スタートアップルーチン 31
B.5 リンカースクリプト 31

C H8 マイコンキットの作り方 33

諸注意

以下，注意事項を箇条書きするので，厳守すること．これは,「ライントレースカーの製作実習」にのみ適用する．

1. 実験時の服装など
   - PC ルームで実験を行うので，清潔な服装であること．作業服の必要はない．
   - 履物は，PC ルーム備え付けのサンダルを使うこと．
2. 実験ノート
   - A4 の実験ノートを用意すること．ルーズリーフは不可である．
   - 実験ノートへの記述は，ボールペンあるいは万年筆とする．鉛筆は不可である．
3. レポートの提出
   - 2 日目の実験日から，毎回提出すること．実験時間中にチェックを行い，不備な点の再提出を求める．
   - 次回の実験日の AM8:45 までに提出すること．それ以降に提出したものは，減点の対象とする．
   - 前期あるいは後期の最後の実験の場合は，その 1 週間後とする．
   - 提出期限に遅れたものは，減点とする．
   - 未提出のレポートがある場合は，単位を与えることができないので注意すること．
   - 提出先は，坂本研究室のポストとする．
4. レポートの再提出
   - 内容に不備があるものはレポートの再提出を課す．
   - 再提出の期限は，再提出を言い渡された 1 週間以内とする．
5. レポートの書き方
   - ガイダンスのときには配布した「実験実習のレポートの書き方」を参考にすること．
6. その他
   - 実験を休んだ場合，放課後を利用して，実験を行うこと．実験を行う場合，機材を貸し出すので，申し出ること．
   - 放課後，実験を行うこともできる．その場合は，担当教員 (坂本) に申し出ること．

実験テーマ 307
ライントレースカーの製作実習

1　目的

制御部にルネサステクノロジ社のマイコン H8 を使ったライントレースカーを作成する．このライントレースカーの作成を通して，センサーからの信号を処理して，機器の制御を行う基本を学習する．

さらに，最終日にコンテストを行う．このコンテストと通して，プロジェクトを企画・運営することを学ぶ．

2　実験のスケジュール・場所

実験は，1 回 150 分を 4 回にわたって行う．スケジュールおよび場所は，表 1 のとおりである．ただし，3 日目の実験終了時刻までにライントレースカーが完成しなかった者は，放課後に，製作を行うことは可能である．その場合は担当教員に申し出ること．

表 1: 実験のスケジュールと場所．場所は，計測:電気計測実験室 (電気棟 1F)，情報: 電子情報工学実験室 (電気棟 3 階の PC ルーム)，ホール:コミュニティホール (電気棟 2F) を表している．

実験日	場所	内容
1 日目	計測	実験のガイダンスを行う．
	計測	機械部品を取り付ける．
	計測	回路を作成し，間違いが無いことを確認する．
2 日目	情報	サンプルプログラムを用いて，ライントレースカーを動作させる．もっとも単純なプログラムで，もっとも低速で動作させることにより，ハードウェアーを確かめる．
	情報/計測	より早くライントレースできるように，プログラムおよび機械部品の工夫を行う．
3 日目	情報/計測	より早くライントレースできるように，プログラムおよび機械部品の工夫を行う．
	電気計測実験室	コンテストの役割を決めて，準備を行う．
最終日	情報/計測/ホール	最終調整
	ホール	各自のライントレースカーを持ち寄って，コンテストを行う．
	ホール	表彰式.

3　原理

センサーの信号に応じて，モーターを制御することにより，線に沿ってライントレースカーを走らせることができる．例えば，図 1 のようにライントレースカーが右にずれたならば，左側のタイアのモーターを減速させて，左の方向に向きを変える．そして，元の位置の戻るようにする．左にずれた場合も，右に向きを変えるようにする．このように制御することにより，ラインに沿ってライントレースカーを走らせることができる．

この実験での制御には H8 と呼ばれるマイコンを使う．センサーの信号をマイコンで読み取り，その内容に従って，モーターを PWM 制御[1]行う．

ライントレースカーの速度は，制御の方法に大きく依存する．低速だと，図 1 のような ON/OFF 制御でもラインに沿って車を走らせることができるが，高速になると困難になる．比例制御あるいは，PID 制御のようなことが必要になるだろう．

諸君は，比例制御や PID 制御はまだ学習していないので，これらを適用することは難しいだろう．諸君の考えたいろいろな制御方法を試して，もっとも高速にライントレースができる方法を探すことになる．プログラムを変更のみで制御の仕方を変えることができるので，いろいろと試してみると良い．

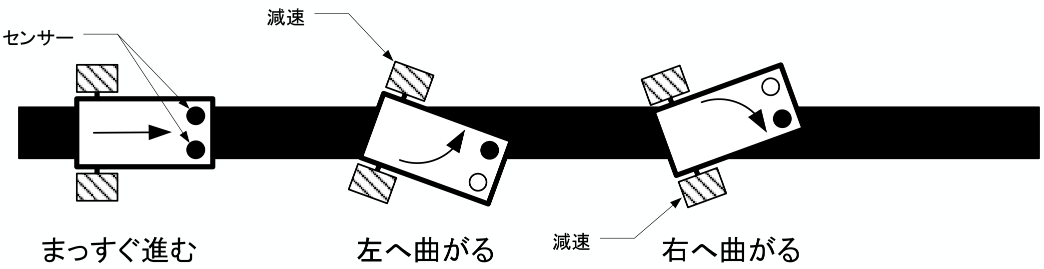


図 1: ライントレースカーの動作原理．センサーがラインを外れるとモーターの速度を変化させて，車の方向を変える．いつも，ラインの方向に車を方向転換させることによりライントレースを行う．

4 実験機材と部品の使用上の注意

4.1 実験機材

ここでの実験は，班でひとつの実験を行うのではなく，ひとりひとりが個別にライントレースカーを製作する．そのため，各人毎に部品を配布する．他の人の部品を流用してはならない．もし，部品の不足があるならば担当の教員に申し出ること．

表 2 に，ひとり当たり必要な実験器材を示す．これらの機材を用いると，基本モデル (低速) あるいは高速モデルのライントレースカーの作成ができる．

プログラムの作成はクロスコンパイラーで行う．Linux がインストールされているパソコンでプログラム作成からコンパイルまで行い，バイナリーデータを H8 マイコンに転送する．電気棟 3F の電気情報工学実験室に設置しているパソコンには，プログラム開発に必要な環境をインストールしてある．

[1]Pulse Width Modulation; パルスのデューティー比（ON と OFF の時間の比）により電流を変化させて，DC モーターの速度をコントロールする．詳細は，前期の実験を見よ．

表 2: 実験器材．基本モデルと高速モデルに対応できるように，機材を準備している．

装置	メーカー	型番/仕様	数量
パソコン			1
H8 マイコンボード	秋月電子通商	AKI-H8/3664N	1
ブレッドボード	サンハヤト	SAD-01	1
ニッケル水素電池		単三型 1.2 [V] 2300 [mA]	4(内予備 2 個)
9V 電池 (006P)		9 [V]	1
赤色 LED			4
トランジスター	東芝	2SC1815	4
ダイオード			2
レンズ付き反射型フォトリフレクタ	SHARP	GP2S05	5
インバーター (NOT 回路)		74HC04AP	1
抵抗		15 [kΩ]	5
		5.1 [kΩ]	2
		3.3 [kΩ]	4
		10 [Ω]	1
タクトスイッチ			1
ジャンパーピン			3
D-SUB コネクター			1
ブレッドボード配線材			1 式
キャスター	ハンマーキャスター	420G-N	1
トラックタイアセット	タミヤ		1 セット
オフロードタイアセット	タミヤ	楽しい工作シリーズ No.96	1 セット
ツインモーターギヤボックス	タミヤ	楽しい工作シリーズ No.97	2
ユニバーサルプレート (2 枚セット)	タミヤ	楽しい工作シリーズ No.157	1
M2.6×10 十字穴付なべ小ねじ			2
M2.6 ナット			2
M2.6 ワッシャ			4
M3×30 十字穴付なべ小ねじ			4
M3×40 十字穴付なべ小ねじ			1
M3 ナット			15
M3 ワッシャ			14

4.2 使用上の注意

実験をはじめる前に以下の注意を読み，正しく機器を使うこと．ここでは，誤った使い方をして，破壊しやすいあるいは危険なものとして，半導体と電池，ブレッドボートについて注意を行う．

4.2.1 半導体

半導体部品は定格を超えた使い方をすると，即座に壊れる．コイルやコンデンサー，抵抗などのパッシブな部品と比較して，その定格の電流や電圧が低いのが普通である．そのため，半導体部品は丁寧に取り扱わなくてはならない．以下，注意事項を列挙する．

- 指定以上の電圧を加えないこと．高い電圧を加えると，半導体は簡単に壊れる．
- LED やトランジスター，H8 マイコンの出力端子には電流制限抵抗を付けて，グランドと接続している．直接電圧を印加すると最大定格以上の電流がながれ，半導体が破壊される．回路図のとおり，抵抗を接続すること．
- LED やダイオード，トランジスターには極性があり，正しく接続する必要がある．図 2 や図 3，図 4 に，それぞれの極性を表す．
- インバーターの電極を図 5 に示す．ひとつのパッケージに同一の 6 個の論理素子が組み込まれている．これも極性を間違えないで接続すること．
- 赤外線センサーの回路の保護のための電流制限抵抗を正しく取り付けること．
- 特に，今回の実験では 9 V 系統と 2.4 V 系統があるので，くれぐれも電圧を間違えないこと．

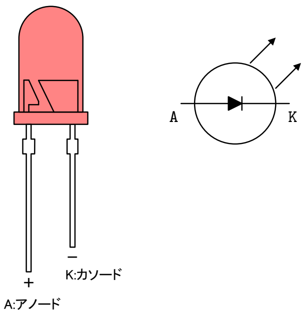


図 2: LED 外観と記号

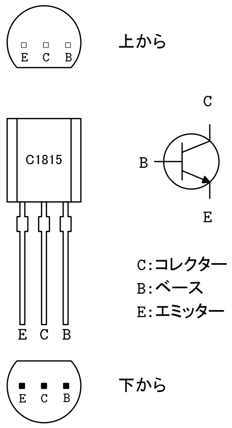


図 3: トランジスター (2SC1815) の外観と記号

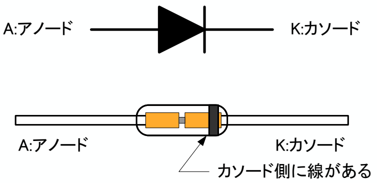


図 4: ダイオードの記号と外観．

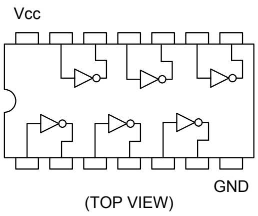


図 5: インバーター 74HC04AP の電極．

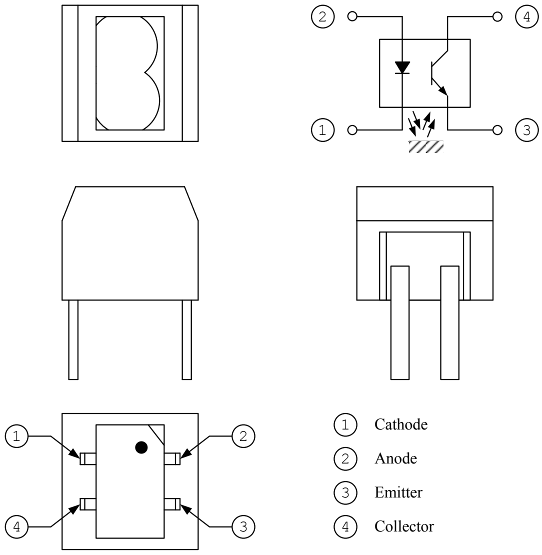


図 6: 反射型フォトリフレクター．

4.2.2 電池

電池は取り扱いを誤ると，非常に危険である．短絡 (ショート) させてしまうと，とても高温になる．なぜならば，一般に電池の内部インピーダンスは低く，多くの電流を流すことができるからである．

細心の注意を払い実験を行わなくてはならない．**電池を絶対にショートさせるな!**

- ライントレースカーの回路の短絡が無いことを確認した後，電池を取り付けろ!
- 次週の実験まで，ライントレースカーに電池をつけた状態のままにするな!
- 電池を片づけるときは，絶対にショートしないように収納しろ!

4.2.3 ブレッドボード

前期の実験と同じように，図 7 に示すブレッドボード上に回路を作成する．ブレッドボードだと，回路の変更が容易である．半田付けを行った場合，回路の変更が大変であることは容易に想像がつく．そのようなことから，頻繁に回路の変更があるような場合には，ブレッドボード上に実験的に回路を作成することがよく行われる．

ブレッドボードを見て分かるように，たくさんの小さな穴が開いている．穴の間隔は 1/10 インチとなっており，それは IC(Integrated Circuit) の電極の間隔に等しい．この穴に IC を差し込んで，回路を作成する．IC に限らず，抵抗やスイッチ，トランジスター等の半導体部品も差し込むことができる．差し込んだ部品は配線材 (ジャンパー線) を使い電気的に接続することで電気回路になる．

ブレッドボードを使うためには，内部の配線を理解しなくてはならない．図 7 に，ここで使うブレッドボードの内部配線をしめす．数個の穴が内部で電気的に接続されている．この接続を理解して，ブレッドボードを上手に使おう．

電子部品と専用の配線材 (ジャンパー線) をブレッドボードに差し込むことにより，電気回路を作成する．その配線材を使うとき，以下を厳守すること．

- 配線材の先端を曲げてはならない．先端を曲げるとすぐに折れてしまう．
- ブレッドボードに差し込むとき，先端をまっすぐにして，真上から差し込むこと．もし，線の先端が曲がっていたならば，まっすぐに直してから，差し込むこと．曲がった配線をブレッドボードに差し込むと，抜けなくなる．あるいは，内部で配線材が切れて，ショートすることがある．
- ブレッドボードの赤いラインに沿った穴は，電源ラインにする．
- ブレッドボードの赤いラインに沿った穴は，グランドラインにする．グランドラインは全ての回路で同電位とすること．すなわち，グランドラインは全てジャンパー線で接続されなくてはならない．

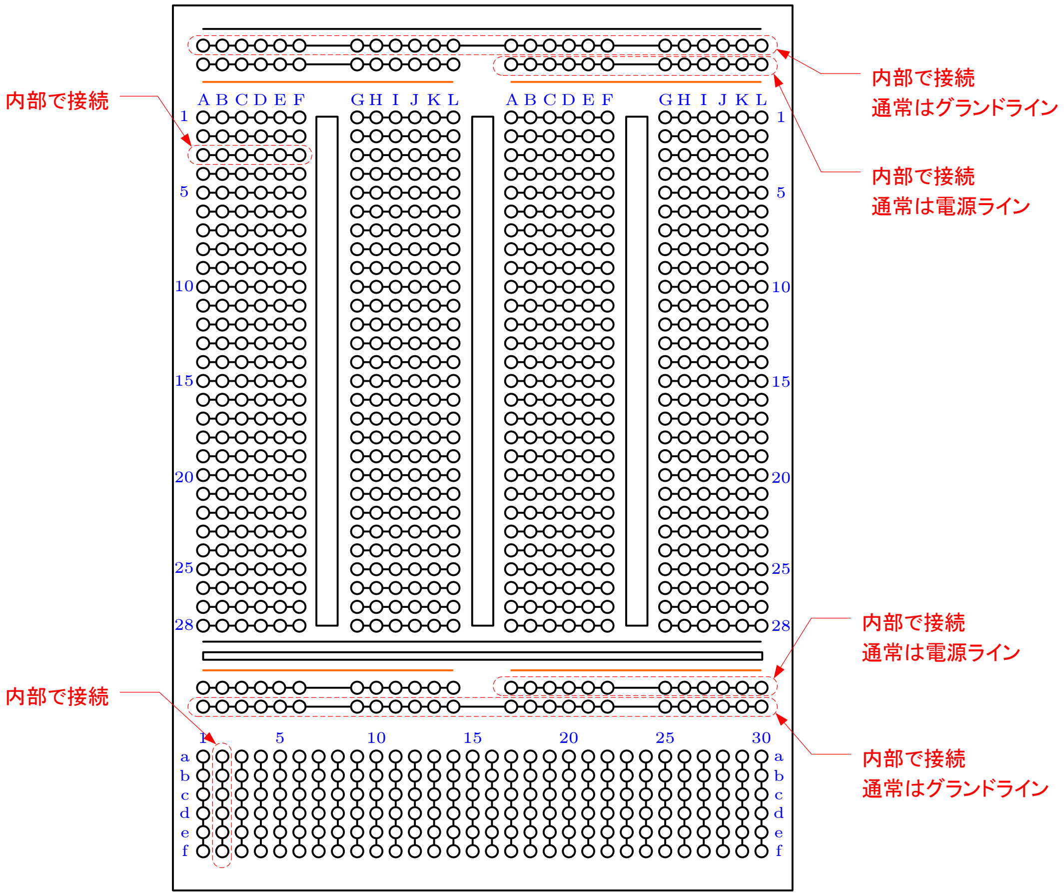


図 7: ブレッドボードの内部配線

5 実験方法

ここでは，実験の手順を述べる．機器の動作やプログラムの詳細については，pp.20–33 に記述している．

5.1 車体の製作

2 枚のユニバーサルプレートを車のシャーシにして，ギヤボックスやキャスター，タイア，センサーなどを図 8 のように取り付け車体を製作する．最初に製作するライントレースカーはここで示す基本モデルとする．基本モデルの動作を確認の後，ライントレースカーレース (秋田高専グランプリ) で勝つことを目指して，高速モデルにチャレンジすることになる．

ギヤボックスは，高速ギヤボックスと低速ギヤボックスの 2 種類用意している．基本モデルでは，低速ギヤボックスを使うこと．低速ギヤボックスは，’C’ のマークの部分に回転軸がある．一方，高速ギヤボックスは’A’である．

車体製作時の注意を次に列挙するので，厳守すること．

- 低速ギアボックスを使うこと．
- 部品をきちんと固定すること．

- 図 8 に示すねじ[2]を用いて，機械部品を固定する．その際，ワッシャー (座金) を必ず取り付けること．これが無いと，穴から外れる場合もあるし，シャーシなどを傷つけることもある．

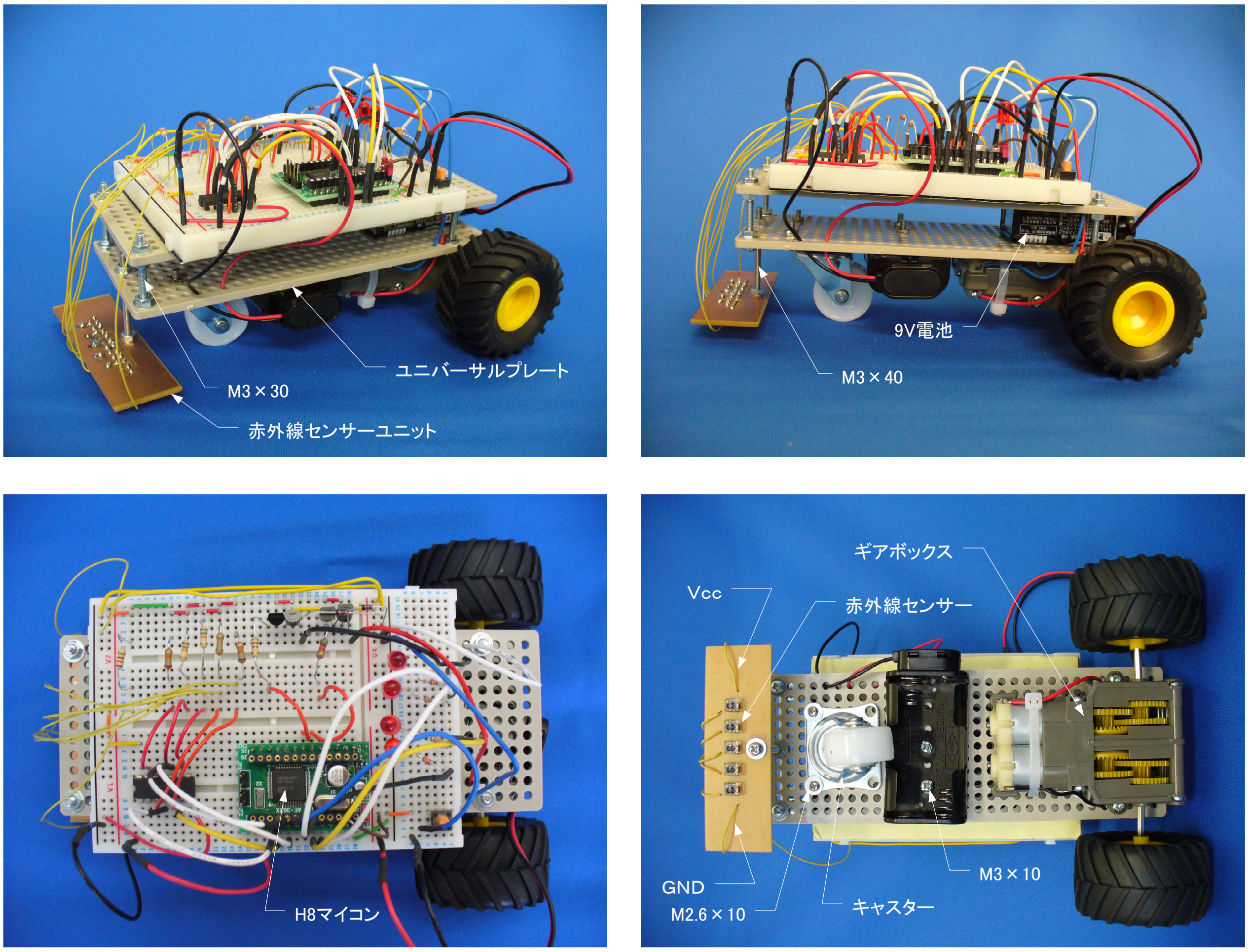


図 8: ライントレースカーの基本モデルの構造

5.2 回路作成

5.2.1 ブレッドボー上の H8 のピン配置

まず初めに，図 8 や図 9 のようにブレッドボード上に H8 マイコンを取り付ける．取り付けの際，方向を確認すること．このとき，JP1 をジャンパーピンでショートさせること．これ以降，この実験ではこの JP1 のジャンパーピンを取り外してはならない．もし，このジャンパーピンを取り外して 9 [V] を印可すると，マイコンが破壊される．

図 9 に H8 マイコンボードのピン番号を示す．H8 マイコンボードの下側の電極はブレッドボードに差し込まれるので，それに接続されているブレッドボードの穴を通して配線を行う．H8 マイコンボードの上側の電極は，直接，配線材を接続する．

[2]ねじに「M2.6 × 10」と書いてあれば，ねじの直径が 2.6 [mm] でなねじの長さが 10 [mm] ということを表す．

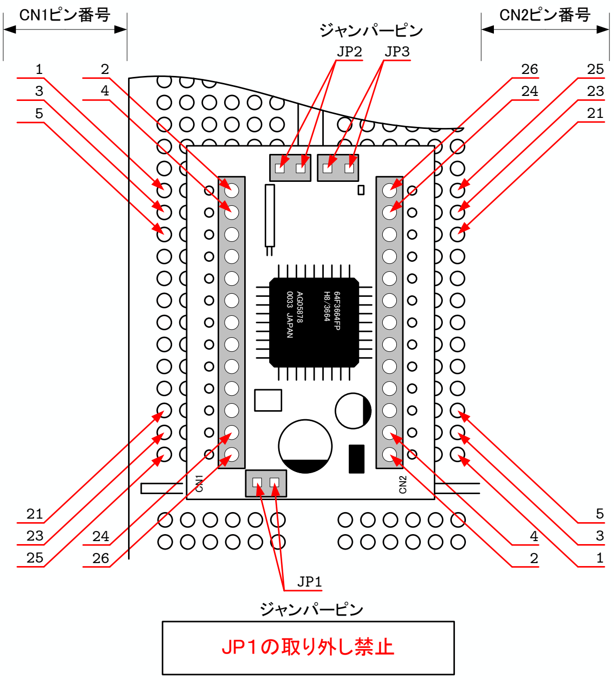


図 9: ブレッドボード上での H8 のピン番号

5.2.2 制御回路

概要

図 11 にライントレースカーの電気回路を示す．主な回路は，ラインを探すセンサー部とその表示 (LED) 部，H8 マイコン，モーター駆動部である．全ての回路はジャンパー線により接続され，その中心に H8 マイコンがある．全ての制御信号は H8 マイコンに入出力され，ライントレースカーの制御が行われる．H8 マイコンの入出力の電極については，p.20 の付録 A.1 「H8 マイコン」で詳しく記述している．

センサー部のフォトトランジスターの出力はアナログなので，インバーター[3](74HC04AP) で信号の出力を二値化—0 と 2.4 [V] に—している．この二値化した信号を H8 マイコンの AD 変換器で読み取っている．AD 変換器で読み取るため，二値化しなくても制御は可能である．ここでは，センサーの推奨回路にしたがい，インバーターを使った．センサーの動作の詳細については，p.20 の付録 A.2「赤外線センサー」に記述している．

DC モーターの制御には Pulse Width Modulation(PWM) を使っている．PWM ではある周期— ここでは 30.5 [Hz] —で，モーターの ON/OFF を繰り返す．ON と OFF の時間の比（デューティー比）を変えることにより，モーターに流れる平均電流を制御する．これにより，モーターの回転数を変化させることができる．センサーの情報に応じて，左右のモーターの速度を変化させることにより，ライントレースカーの方向制御を行っている．モーターの構造と PWM 制御については，p.21 の付録 A.3「DC モーターと PWM 制御」にもう少し詳しく記述している．

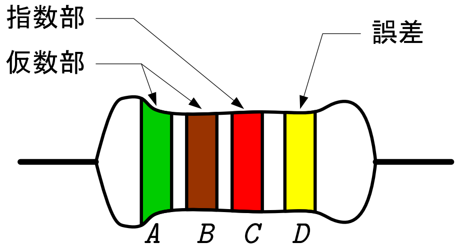
モーターのスイッチングには，ダーリントン接続[4]したトランジスターを用いている．トランジスターひとつでは増幅率が不足しているからである．ここで使うトランジスターひとつの増幅率は 100 倍程度であるが，モーターを駆動するためには 200 倍程度の増幅率が必要となる．H8 マイコンからの出力の電流は最大でも 1 [mA] 程度で，DC モーターでは 200 [mA] 程度の電流が必要で，200 倍程度の増幅率となる．トランジスターの動作については，p.23 の付録 A.4「トランジスター」に更に詳しく説明している．

[3] 論理回路の NOT
[4] ダーリントン接続すると，増幅率は一つのトランジスターの 2 乗になる．

モーター電極の両端には，フリーホイールダイオードを接続している．このようにすることにより，PWM の OFF 時にコイルに発生する逆起電力による悪影響を防止するとともに，それをモーターの回転エネルギーに変えている．動作の内容については，p.24 の付録 A.5「ダイオード」に更に詳しく説明している．

回路製作　ここでは，ライントレースカーの電気回路 (図 11) を作成する．p.7 の図 8 を参考に，H8 マイコンや LED，半導体部品などを接続すること．以下，注意点を列挙しておく．

1. 回路を作成しているとき，部品の破損を防ぐために電源を接続していはならない．意図しない電流が素子に流れることがあるからである．
2. 初めに，ブレッドボード上に電源とアースのラインを作成する．まだ，電源を接続しない．
3. ブレッドボード上に必要な部品を取り付け，配線材 (ジャンパー線) で接続する．
   - 配線材の先端を曲げてはならない．
   - 配線材は，まっすぐ真上から差し込むこと．
4. センサーやモーターの配線をを行う．回転部品に巻き込まないように配線すること．
5. 抵抗はカラーコード (図 10 を参照) を読み，テスターでチェックした後，接続すること．
6. 回路が完成したならば，極性および接続に間違いの無いことをチェックする．



抵抗 $R=(10A+B)\times 10^{C}\pm D$

(例) 緑茶赤金

$R=(50+1)\times 10^{2}\pm 5\%$

精度5%の5.1[kΩ]の抵抗

	A	B	C	D
黒	0	0	0	—
茶	1	1	1	—
赤	2	2	2	—
橙	3	3	3	—
黄	4	4	4	—
緑	5	5	5	—
青	6	6	6	—
紫	7	7	7	—
灰	8	8	8	—
白	9	9	9	—
金	—	—	-1	5%
銀	—	—	-2	10%

図 10: 抵抗のカラーコードの読み方

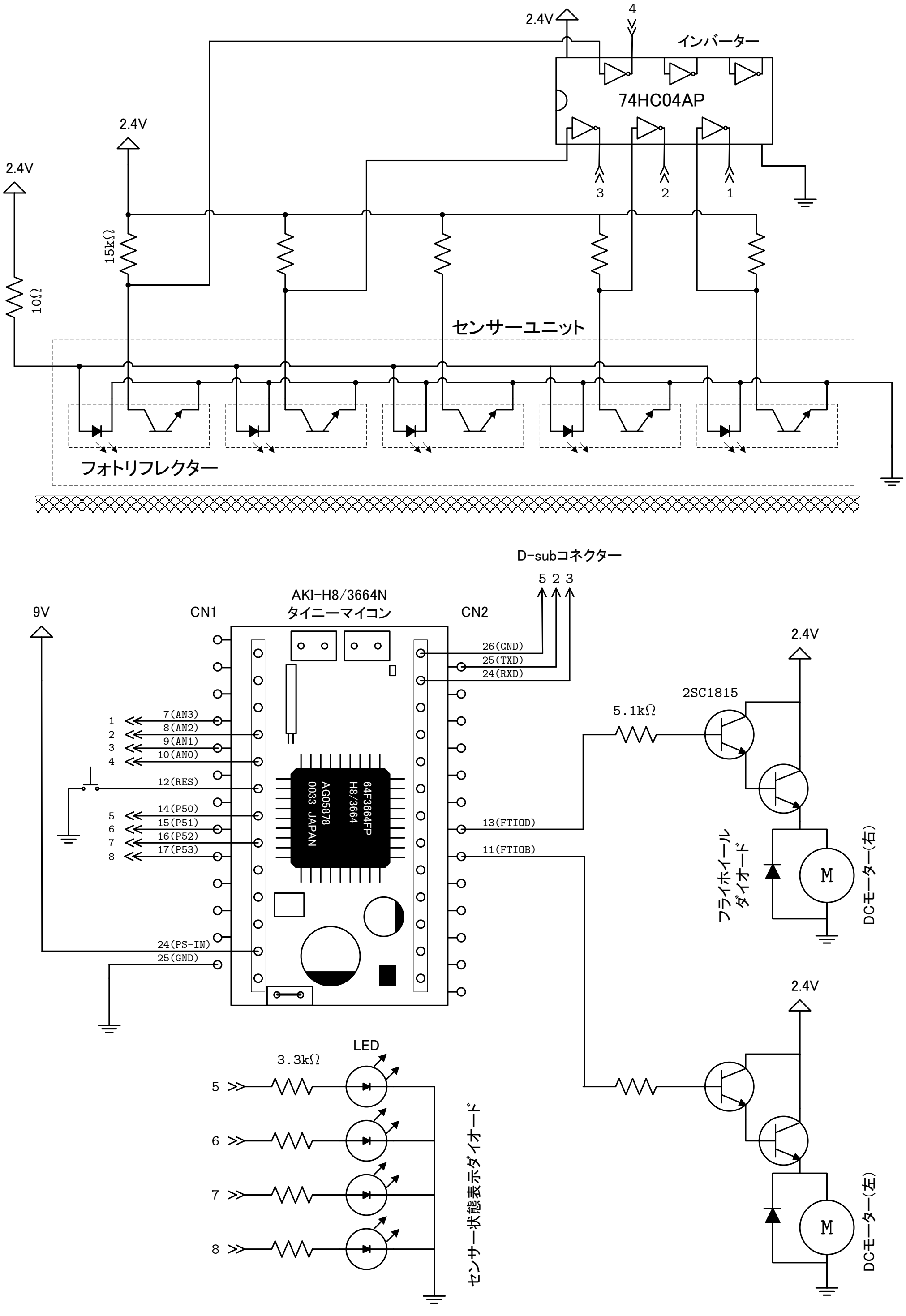


図 11: ライントレースカーの回路

5.3 ハードウェアーの確認

全ての部品を組み付け，回路ができあがったならば，以下のことを確認すること．

- 部品がきっちりと固定されていることを確認すること．ねじなどがゆるんでいる場合は，締め直すこと．

- センサーの高さが，3〜5 [mm] であることを確認すること．センサーの焦点距離は 4 [mm] であり，大きく外れると感度が低下する．
- タイアが回転した場合，電線を巻き込まないように配線ができているか?—確認すること．
- 図 11 の通りに，回路が構成されていることを確認すること．以下の項目を確認し，回路図にチェックマークを記入すること．
  - 回路の配線の経路を確認する．
  - グランド (0 [V]) が共通となっている—ことを確認する．特に，9V の電池と単三電池二本の 0 [V] が接続されているを確認する．
  - H8 マイコンの JP1 のジャンパーピンが接続されている—ことを確認する．
  - トランジスターの型番と接続方法が正しい—ことを確認する．
  - カラーコード (p.9 の図 10) を見て，全ての抵抗の値を確認する．
  - LED の接続と方向を確認する．
  - ダイオードの接続と方向を確認する．
  - インバーターの接続の接続を確認する．
  - H8 マイコンの電極の接続を確認する．
  - リセットスイッチの接続を確認する．
  - センサーユニットの接続を確認する．

5.4 プログラム作成

ハードウェアーの確認が終われば，電気情報工学実験室でプログラムの作成を行う．

5.4.1 ソフトウェアーについて

この実験のプログラムは，Linux の gcc を使って開発する．gcc は統合開発環境ではないため，プログラム開発は原始的な部分[5]から作成しなくてはならない．原始的な部分からプログラムを開発すると大変ではあるが，コンピューターに関する多くの知識が得られる．

この実験では原始的な部分に触れないが，勉強は可能である．この実験で使う全てのプログラムを，pp.26–33 の付録 B に載せておく．このような原始的なプログラムに，興味があれば参考文献 [1] から読みはじめればよいだろう．

この実験で必要な原始的な部分のプログラム (付録 B) は全て，実験実習の WEB ページ[6] からダウンロードして使う．これらのプログラムの中身を全て理解できれば，一人前である．興味のある者はこれを理解するよう努力せよ．コンピューターの仕組みがよくわかるようになる．

5.4.2 制御プログラムの作成と転送

ライントレースカーレース (秋田高専グランプリ) に向けて高速化の工夫を行う前に，確実に動作するプログラムで，マイコンの動産の確認を行う．

注意事項　プログラム作成に先立って，以下の注意事項を読み，厳守すること．

- H8 マイコンへのプログラム転送の手順を間違えないこと．
- プログラムが暴走したら，`[Ctrl]` に続けて `[c]` を押すことにより，強制終了させる．

[5] アセンブラやリンカースクリプト，スタートアップルーチンを指す．
[6] http://akita-nct.jp/saka/lecturenote/3e/3e.html

プログラムの作成 プログラム作成には Linux を用いる．電気情報工学実験室のパソコンには，H8 マイコンを開発するためのプログラム類がインストールされている．

以下の順序に従い，もっとも基本的なプログラムを作成せよ．

1. D-SUB コネクターを使って，パソコンとライントレースカーを接続する．ただし，この状態では，ライントレースカーの電源 (2.4 [V] と 9 [V]) は OFF にしておく．
2. パソコンの電源を ON にして，Linux を立ち上げる．
3. プログラムのソースファイルを保存するディレクトリーを作成する．ディレクトリー名は，全てアルファベットで記述すること．
4. プログラムに必要な以下のファイルを実験実習の WEB ページ[7]からダウンロードし，先ほど作成したディレクトリーに保存する．

`Makefile`	コンパイルの方法や H8 への転送方法を記述している．
`3664.h`	主にレジスターのアドレスを記述している．
`h8c.c`	ライントレースカー用の関数．
`h8c.h`	ライントレースカー用の関数のヘッダーファイル．
`h8_start.s`	スタートアップルーチン．
`h8_link.x`	リンクの方法を記述しているリンカースクリプト．

   ファイル名や拡張子を変えてはならない．Makefile をダウンロードするときに余分な拡張子—.htm や.html—が付くことがあるので，取り除くこと．

5. リスト 1 のとおり H8 マイコンのプログラムを作成する．作成したファイルは，先ほどダウンロードしたファイルと同じディレクトリーに保存すること．
   - ファイル名は，「`experiment.c`」とする．
   - コメント文は，記述しなくてもよい．
   - ダウンロードしたファイルは，変更してはならない．
6. コンパイルする．コマンドは,「`make`」である．
7. `make` の結果，できあがったプログラムを H8 マイコンへ転送する．
   (a) H8 マイコン基板の JP2 と JP3 をジャンパーピンでショートする．JP1 は常にショートしておくこと．
   (b) 回路に 9 [V] を供給する．9 [V] 電池をブレッドボードに接続する．
   (c) コマンド「`make write`」をタイプし，プログラムを転送する．転送がはじまると，「`H8/3664F is ready!  2001/2/1 Yukio Mituiwa.`」と表示される．もし失敗したならば，`[Ctrl]` と `[c]` を押して，プログラムを止める．
   (d) 転送には，20 秒くらい必要である．「`EEPROM Writing is successed.`」と表示されるまで待つ．

リスト 1: 基本プログラム．もっとも単純なライントレースカーの制御プログラム

```
#include "3664.h"
#include "h8c.h"

int main()
{
  init_led();
```

[7] http://akita-nct.jp/saka/lecturenote/3e/3e.html

```
  init_pwm();
  init_ADC();

  duty_pwm(0,0);   // PWM の初期値
  start_pwm();     // PWM 信号出力の開始


  while(1){
    ADC();         // AD 変換器でセンサーデータ取得

    //---- LED の発光 -----
    PDR5=((0x01*ch1+0x02*ch2+0x04*ch3+0x08*ch4) & 0x3f) | (PDR5 & 0xc0);

    //---- DC モーターの制御 -----
    if(ch2==1 && ch3==0) duty_pwm(0,100);
    if(ch2==1 && ch3==1) duty_pwm(100,100);
    if(ch2==0 && ch3==1) duty_pwm(100,0);

  }

}
```

5.5 実行

まずは，転送したプログラムを実行させて，ライントレースカーの動作を確認する．動作確認の順序は，以下の通り．

1. マイコンに電力を供給している 9 [V] の電池を OFF にする．ブレッドボードから 9 [V] のジャンパー線を引き抜く．
2. H8 マイコン基板の JP2 と JP3 のジャンパーピンを取り外し，オープンにする．決して，JP1 を取り外してはならない．
3. ブレッドボードに，モーターおよび赤外線センサー駆動用のニッケル水素電池 2 本の 2.4 [V] を供給する．
4. マイコンに 9 [V] の電池を ON にする．ブレッドボードから 9 [V] 電池のジャンパー線を取り付ける．
5. すると，プログラムが実行される．
6. プログラムを最初から実行させたければ，RES スイッチを押す．

次に，練習用あるいはグランプリ用のサーキットにライントレースカーを持ち込んで，ちゃんと黒線上をトレースすることを確認する．ライントレースができることが確認できれば，ハードウェアーは完成である．

5.6 改良

ライントレースカーを高速化するためには，ハードウェアーとソフトウェアーの改良が可能である．

5.6.1 ハードウェアーの改良

ギヤボックスやタイア，前輪はそれぞれ 2 種類用意している．基本モデルを改良するとき，これらの部品を上手に選定し，ライントレースカーの速度の向上を図ることができる．高速になった分，制御は難しくなるので，プログラムの改良も必要であろう．

高速化のためにモーター駆動用の電池の電圧を上げることは，禁止する．基本モデル同様に，単三型のニッケル水素電池 2 個を使用すること．電圧をあげることは，高速化には大変有効な方法である．しかし，他の半導体部品を損傷する可能性があるため，ここでは禁止する．

ギヤ比の小さい高速のギヤボックスが使用可能である．これを使うことにより，より高速にライントレースカーを動かすことができる．ただし，高速に動作する分，制御が格段に難しくなる．

タイアを変えることにより速度を変えることもできる．大小 2 種類のタイアを用意しているので，上手に選択せよ．

高速化のためには，4 個のセンサーを上手に使う必要がある．ラインを探す赤外線センサーのユニットにはセンサーが 5 個組み込まれている．しかし，H8 マイコンの入力は 4 個である[8]．この 5 個の内，どの 4 個を使うかは諸君の考え方次第である．また，基本プログラム (p12 のリスト 1) では 2 個のセンサーのみで制御を行っている．多くのセンサーを制御に使えば，より高速にライントレースカーを動かすことができるであろう．

また，センサーやタイアの位置を変えることでも高速化ができるだろう．センサーやタイアの位置を変えて，もっとも制御しやすい位置を探すと良い．

5.6.2 ソフトウェアーの改良

制御方法を変えるだけでも，ライントレースカーは高速に動作する．諸君が制御できるのは，センサーの入力と DC モーターの速度，LED の点灯である．

センサーの入力信号は，グローバル変数 `ch1`〜`ch4` に設定される．`ch1`〜`ch4` のそれぞれのグローバル変数は，H8 マイコンボートの `AN0`〜`AN3` の入力に対応する．もし，接続されているセンサーが白を検知したならば，グローバル変数の値は整数の 0 となる．黒ならば，整数の 1 になる．AD 変換器を実行させる関数 `ADC()` を呼び出したとき，これらのグローバル変数の値が設定される．

センサーの状態は LED により，観測することができる．センサーの状態を表すグローバル変数 `ch1`〜`ch4` の値を，レジスター `PDR5` に次のように設定する．

```
PDR5=((0x01*ch1+0x02*ch2+0x04*ch3+0x08*ch4) & 0x3f) | (PDR5 & 0xc0);
```

すると，センサーが黒を感知した場合，LED が点灯する．白の場合は消灯である．

ライントレースカーを制御するために，次の関数を用意している．いずれも，基本プログラム (p.12 のリスト 1) に含まれているので，使い方は容易に分かるであろう．これらの関数の詳細を知りたければ，p.27 のリスト 4 を見よ．

void **init_led**(void)
: LED 表示のための初期化関数．センサーの状態を LED で表示させ得るために，最初に 1 度呼び出す必要がある．LED はレジスター `PDR5` の下位 4 ビットを表示している．

void **init_pwm**(void)
: PWM 出力のための初期化関数．PWM 出力を使ってモーターを制御するときに，最初に 1 度呼び出す必要がある．

void **duty_pwm**(unsigned int *b*, unsigned int *d*)
: この関数を呼び出すと，H8 マイコンから出力される PWM のテューティ比を変えることができる．第一引数の b は FTIOB 端子，第二引数の d は FTIOD 端子のディーティ比を決める．両方とも 0〜100 の値の整数を引数とし，デューティ比 [%] を示す．引数の値が 0 の時にデューティ 0 [%] となり，モータは停止する．100 とすれば，最高速度でモーターは回転する．この関数を使う前に，`init_pwm()` を事前に一度呼び出しておく必要がある．そして，この関数でデューティ比を設定した後，`start_pwd()` 関数により PWM 信号が出力される．その後は，この関数によりデューティ比を変えれば，即座に PWM 出力が変わる．

void **start_pwm**(void)
: モーターを制御するための PWM 信号を出力する．一度呼び出せば，デューティ比は，`dugy_pwm()` 関数を呼び出す毎に変化させることができる．

[8]H8 マイコンは 8 入力も可能であるが，プログラムが面倒だったので 4 入力とした．

void **init_ADC**(void)
: 赤外線センサー信号入力のための初期化関数．実際には H8 マイコンの AD 変換器を設定して使用可能な状態にしている．赤外線センサーの信号処理のために，最初に 1 度呼び出す必要がある．

void **ADC**(void)
: 赤外線センサーの信号を取り込みを行う関数．この関数が呼び出される毎に，センサーの状態を表すグローバル変数 `ch1`〜`ch4` の状態が変化する．

もし，これらの命令で不足と感じたならば，新たに独自の関数を追加することも可能である．自信のある者はトライせよ．

5.7 秋田高専グランプリ

実験の最終日，ライントレースカーの競技大会「秋田高専グランプリ」を行う．そこでは，各自工夫したライントレースーを持ち寄り，レースを行う．レースは図 12 のサーキットで行う．このサーキットを上手にトレースした車が勝者となる．

トレースする黒い線は，幅 23 [mm] である．センサーの間隔は，この線の幅を考えて設定している．

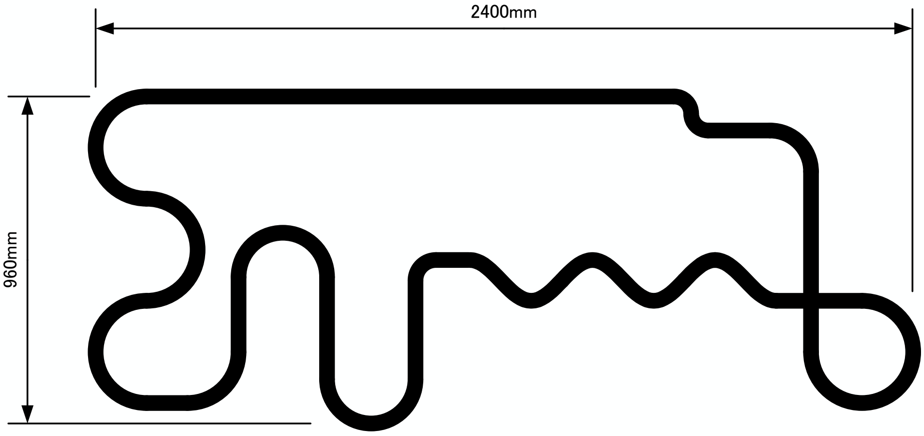


図 12: 秋田高専グランプリのサーキット．線幅は 23 [mm] である．

この競技の準備および運営も実験の一部である．グランプリのルールと各自の役割を決め，自主的にグランプリを運営することを求める．例えば次のような役割が必要であろう．

- 運営委員会
- 準備係
- 記録係
- タイム測定係
- その他

このようなプロジェクトを遂行ことは，エンジニアーの中心的な仕事の一つである．プロジェクトを企画し，それを実行することを学習する．

6 考察課題

次に述べる考察課題の内容を調べて，レポートにまとめよ．

[問 1] この実験でのセンサーの以下の動作を述べよ．

- ラインを見分ける方法
- フォトトランジスターの動作
- 論理回路 (NOT) の動作
- H8 マイコンへの入力までの信号の流れ

[問 2] PWM 制御で DC モーターの速度制御ができる理由を述べよ．

[問 3] ダイオードの電極は，Cathode と Anode と言う．名前の由来を調べよ．

[問 4] トランジスターの電極は，Emitter と Collector，Base と言う．名前の由来を調べよ．

WEB や書籍で調べることにより，これらの課題の解答が得られる．レポートに調べた内容を記述する場合，WEB や書籍のコピー (丸写し) は厳禁である．調べた内容を，自分の文章でまとめなくてはならない．自分の文章にまとめることが，訓練になる．

7 レポート作成上の注意

7.1 書き方の注意

心がまえ 学生実験のレポートは，将来，諸君が仕事をするときの報告書作成の訓練のためにある．技術者は，さまざまなドキュメント—報告書，仕様書，計画書，提案書など—を作成する．ドキュメント作成は，技術者にとって最も重要な仕事のひとつである．そのため，技術者教育を行う学校では，ドキュメント作成能力の育成に十分な時間をかけている．

レポートは，読者を想定して書かなくてはならない．自分の伝えたい内容が，読者に分かりやすく伝わる必要がある．諸君が作成する実験のレポートの読者は，指導教員である．しかし，諸君は,「指導教員」に対して報告書を書くのではなく，将来の「上司」あるいは「客先」,「一般の消費者」を想定しなくてはならない．全てが分かっている「指導教員」を想定の読者とすると，あまり訓練にならない．そこで，諸君が作成する学生実験のレポートでは，実験内容を全く知らないが，ある程度の技術的素養のなる「他高専の学生」とするのが良いだろう．これは，丁度，会社の上司や同僚に報告する場合と似た状況である．

分かりやすいレポートを書くためには，**読者の立場に立って考える**ことが重要である．読者は何を知りたいのか? 自分が伝えたいことをどうすれば分かりやすく伝えることができるか?—をよく考えて，内容を吟味し，表現を工夫してレポートを記述しなくてはならない．この場合，想定している読者の知識も考慮しなくてはならない．自分が書いたレポートを「他高専の学生」という立場で読み直し，推敲を重ねると良いレポートができる．

文章 実験のレポートに限らず，一般的な文書は文 → 段落 → 節 → 章というような階層構成となっている．教科書を見よ，この通りになっているはずである．このように記述すると，内容が整理され，読みやすい文書になる．実験のレポートもこの階層構造を守らなくてはならない．

実験実習のレポートを構成する文や段落，節，章の書き方を説明する．詳細については，参考文献 [2] を読むことを勧める．

文 (sentence) 主語と述語があり，単語が集まって文ができる．良い文を書くためには，

- 一つの文で複数のことを述べない．
- できるだけはっきりと分かりやすい文を書く．あいまいな表現は避ける．

というようなことに心がけるべきである．

段落 (paragraph) 文が集まり，段学ができる．段落を作成するためには，

- その段落内のトピックの内容を表す一つの文 (トピックセンテンス) を書く．それも，できるだけ段落の先頭に書く．
- 引き続く文は，トピックセンテンスに関連した内容でなくてはならない．
- 一つの段落内で記述するトピックは一つにする．

というようなことに気をつけるべきである．

節 (section) 段落が集まり，節ができる．章を変えるほどではないが，大きく話題が変わるときには，別の節にして，新たにタイトルをつける．

章 (chapter) 節が集まり，章ができる．章の構成方法は，次節で述べる．

普通，学生実験程度のレポートでは，章はなく，節からはじまる．節が，節 → 小節 → 小々節 → 小々々節 → ··· となっている．

諸君のレポートの多くの問題は，段落ができていないことである．一つの節が文 (1 行) となっていたり，箇条書きしか書いていないものも多い．いくつかの段落があって，はじめて節となり，文章と呼ばれるものになる．将来のために，諸君は段落のある文章を書く努力を行わなくてはならない．

ここの実験レポートでは，段落からなる節をきちんと書けるようになることを目指す．そのためにも，毎回のレポートの提出を課し，添削を行う．

7.2 記述内容

レポートを書く場合，章立てが最も重要である．章立てが良いと，読者は読みやすいし，記述する方も考えがまとめやすい．ようするに，整理された文書ができあがる．

文書によって，章立てはほぼ決まっている．会社で使う報告書や仕様書の章立てはその会社でほぼ決まっている．また，論文の章立ても学会によってほとんど決まっており，どれも似たような構成となっている．学生実験の章立ても，大体の定型がある．この実験では，次の示す章立て—実際には節—で，そこで述べている内容についてレポートに記述しなくてはならない[9]．あわせて注意事項も記述しているので，レポート作成時には，これを守らなくてはならない．

1 目的

どのような目的でこの実験を行ったか?—について書く．実験目的が，書かれていれば，それに引き続く内容が理解しやすい．ここの内容は，指導書に書かれている「目的」の通りに記述すればよい．

2 原理

ここでは，ライントレースカーの大まかな原理を記述する．大まかな原理を最初に説明されると，読者は以降の詳細な内容が理解しやすくなる．以下に注意して，原理を記述すること．

- 概ね指導書の通りに記述しても良いが，指導教員が学生向けに記述した内容は，書き換える．あるいは，削除する．

3 ライントレースカーのハードウェアー

ライントレースカーのハードウェアーについて，その構成と動作が分かるように文章と図を使って説明する．読者が知りたい内容は，大体次のようなことである．

[9]本来，章立て，およびその内容はレポート作成者が考えることである．しかし，諸君はそこまで訓練が行われていないので，ここでは細かく指導する．

- どのような部品や素子でライントレースカーができあがっているのか?
- それがどのように動作して，ライントレースを行うのか?
- どのような回路になっているのか? それがどのように動作するのか?

その他，以下に注意すること.

- 実験機材の表を記述する必要は無い．本文中でこの表を引用することは無いからである．書きたければ，付録に書く.

4 ライントレースカーの制御

どのようにして，黒い線を検知し，ライントレースをするのか?—を記述する．センサーの入力からモーターの制御までの流れを，文章で記述する．さらに，指導書のプログラムのリスト 1 を引用して，動作を説明する.

5 実験結果

高速化に関して自分で工夫した内容を記述する．少なくとも，以下の内容を記述しなくてはならない.

- ハードウェアーで工夫した内容．ハードウェアーについて工夫が無ければ，記述の必要はない.
- 自分で工夫したソフトウェアーのリストとその説明.
- 高速化の結果.

6 考察

この実験を通して，自分で考えたことを記述する．例えば，次のようなことを書く.

- コンテストで優勝したマシンと自分のマシンの差．もし，自分のマシンが優勝したならば，何が他のマシンとの差となったか?
- 自分のマシンの問題点.
- コンテストや様々な実験を通して，より高速のライントレースカーを作成するための方法.

7 考察課題

問いに関して，自分で調べて，きちんとした文章で記述する．箇条書きのみ，図のみ，あるいは 1 文 (sentence) のみではダメ．WEB や文献のコピー (丸写し) は厳禁とする．もちろん，語尾のみを変えるのもダメ．調査した結果をまとめ，自分の文章に直して記述しなくてはならない.

8 感想

この章のみ想定する読者を「指導教員」とし，諸君の素直な感想を書く．加えて，目的で記述した学習内容を鑑みて，実験を通して得られた内容も述べなくてはならない．更に，この実験をよりよいものに変える方法–など積極的な意見を書くとなお良い.

章立てをはっきりさせるためには，見出し (節のタイトル) のフォントを変えると良い．指導書では，全ての見出しはブロック体とし，サイズも変化させている．諸君の提出する実験のレポートも見出しのフォントとサイズを変化させて，読者に分かりやすくなるように工夫しなくてはならない.

7.3 提出要領

この実験は，4 回 (150 分 ×4) で行うが，レポートは 2 回目以降，必ず提出すること．各回のレポートの内容は，表 3 の通りとする．レポートの章立ても，この表の通りにすること.

表 3: 実験回数毎のレポートの内容

実験回数	レポート内容
2 回目	1. 目的
	2. 原理
	3. ライントレースカーのハードウェアー
	7. 考察課題
3 回目	4. ライントレースカーの制御
4 回目	5. 実験結果
1 週間後	6. 考察
	8. 感想

A ライントレースカー構成部品の動作

A.1 H8 マイコン

ライントレーサーの制御には，ルネサステクノロジー社のマイコン H8/3664N を使っている．このマイコンには，様々な機能がある．ここでは，センサーの出力を A/D 変換器で読み込み，その結果を出力端子に接続した LED を用いて表示している．さらに，PWM の信号を出力し，それをトランジスターの ON/OFF 信号に使うことにより DC モーターを制御している．

実験では，H8 マイコンを動作させるための周辺機器を組み込んだ CPU ボード (秋月電子 AKI-3664N) を使う．周辺機器をボードに組み込むことにより，ユーザーの利便を図っている．表 4 に，実験で用いている端子と機能を載せておく．

表 4: 秋月電子の AKI-3664N の端子と機能．この実験で使う端子のみを記述．

端子台	ピン番号	3664N 名称	機能
CN1	7	AN3	A/D 変換器のアナログ入力端子 3
	8	AN2	A/D 変換器のアナログ入力端子 2
	9	AN1	A/D 変換器のアナログ入力端子 1
	10	AN0	A/D 変換器のアナログ入力端子 0
	12	RES	グランド電圧にするとリセット．プログラムは最初から実行．
	14	P50	PDR5 レジスターの最下位ビットのが出力される (0:0 [V] 1:5 [V])
	15	P51	PDR5 レジスターの下位から 2 番目のビットのが出力される．
	16	P52	PDR5 レジスターの下位から 3 番目のビットのが出力される．
	17	P53	PDR5 レジスターの下位から 4 番目のビットのが出力される．
	24	PS-IN	電源入力．JP1 を未接続:5 [V] JP1 をジャンパー 7–12 [V] .
	25	GND	グランド電圧．ここの電位がグランド電位の基準となる．
CN2	11	FTIOB	PWM 信号出力．
	13	FTIOD	PWM 信号出力．
	24	RXD	PC との通信につかう．
	25	TXD	PC との通信につかう．
	26	GND	PC との通信につかう．

A.2 赤外線センサー

赤外線センサーは，ラインを判定するための感知装置である．ここでは，センサーの先が白/黒の判断を行う．

赤外線センサーは，赤外線を発生する LED とそれを関知するフォトトランジスターから構成されている．具体定期な動作を図 13 を使って説明しよう．LED に電流を流すと赤外線を発生する．レンズを通過した後，この赤外線はライントレースカー下の床に当たる．もし床が白ければ (図 13 の左側) 赤外線は反射され，フォトトランジスターに入る．この場合，フォトトランジスターのエミッターとコレクターの間に電流が流れる．一方，床が黒いとき (図 13 の右側) には，赤外線は反射されず，フォトトランジスターには光が入らない．この場合には，フォトトランジスターには電流が流れない．

フォトトランジスターに電流が流れる場合，その抵抗は小さいので，コレクターの電圧は，ほぼ 0 [V] になる．図中のコレクターに接続された抵抗に比べて，フォトトランジスターの抵抗が無視できるくらい小さいからである．一方，赤外線が反射されないでフォトトランジスターに電流が流れない場合，コレクターの電圧は大体

2.4 [V] となる．このとき，コレクターに接続された抵抗に比べて，フォトトランジスターの抵抗がかなり大きいためである．

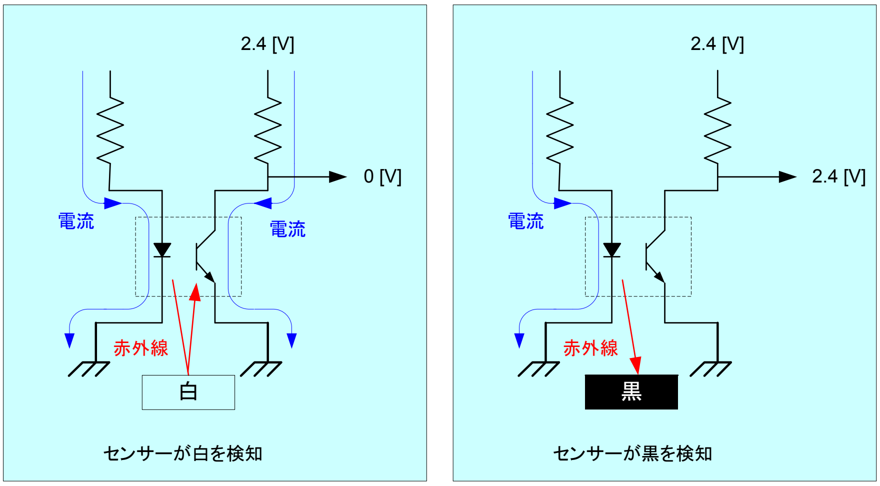


図 13: ラインが黒と白の場合の赤外線センサーの動作.

このフォトトランジスターの動作は，図 14 のように考えることができる．丁度，反射してくる赤外線によりスイッチが ON/OFF しているのである．そうして，スイッチの電源側の電圧を測定すれば，スイッチの ON/OFF が分かる → 赤外線の反射の有/無が分かる → 床の白/黒が分かる．

スイッチの電源側の電圧は，H8 マイコンの A/D 変換器で測定することができる．しかし，ここの実験では，ON/OFF の信号を二値化するために，インバーター; 論理回路の NOT を介して，A/D 変換器へ入力している．

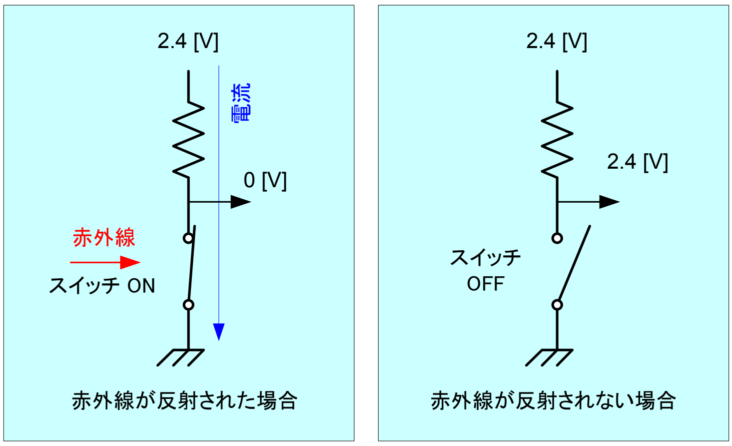


図 14: フォトトランジスターの動作.

A.3　DC モーターと PWM 制御

A.3.1　DC モーターの構造

ライントレースカーの駆動には 2 個の DC モーターを用いている．DC モータは，図 15 に示すとおり，固定子に永久磁石，回転子にコイルを用いている．この回転子のコイルに電流を流すことにより磁場を発生させ，これと永久磁石の磁場との反発力/吸引力によりトルクが生じる．$2\pi/3$ [rad] 回転すると，ブラシに接触している

整流子の電極が変わり，磁場の位相が $2\pi/3$ [rad] 変化する．これが連続して起きるので，直流電源で回転を得られる．

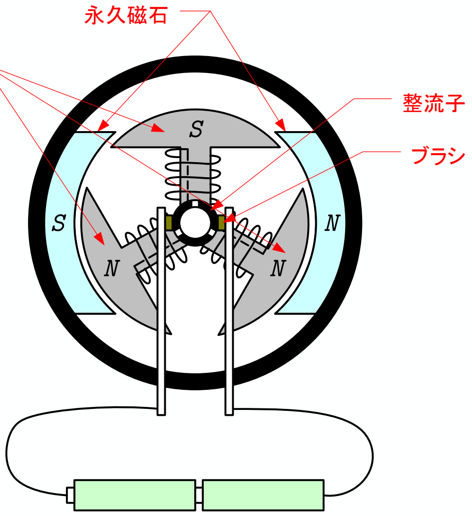


図 15: DC モーターの原理図．直流電流を流すことにより，コイルに磁場が発生し，永久磁石との反発/吸引力により回転する．$2\pi/3$ [rad] 回転すると，整流子の電極が変わり，磁場の位相が $2\pi/3$ [rad] 変化する．これが連続して起きることによりモーターが回転し続ける．

この DC モーターは，次のような長所と短所を持っている．ここで述べている短所は，全てブラシと整流子の接触部分で起きている．簡単な回路で直流モーターを作ると避けることができないことである．

- 長所
  - 起動トルクが大きい．
  - 印可電圧に対して，回転数が比例する．
  - 駆動電流に対して，トルクが比例する．
  - 効率が良い．
  - 安価である．
- 短所
  - 整流子とブラシが機械的に接触して回転するので，電気ノイズが発生する．
  - 機械接触して，絶えずこすれているので，整流子に寿命がある．

このような特徴を生かして，小型のモーターに使われることが多い．諸君は，ミニ四駆で使ったことがあるだろう．

A.3.2 DC モーターの PWM 制御

DC モーターは印可する電圧を変えることで，速度の制御ができる．もちろん，電圧に応じて駆動電流も変化するので，トルクも変わる．しかし，駆動電源の電圧を変化させることは，簡単なことではない．とくに，このライントレースカーのように複雑な回路を避ける必要があるときには電圧の変化による速度制御は避けるべきである．

それに対して，Pulse Width Modulation(PWM) 制御は簡単な回路で，DC モーターの速度制御ができる．PWM 制御では，図 16 のように信号が H になったときにモーターに電流が流れ回転のトルクが生じる．L の時

は，電流が流れないのでトルクは発生しない．この H と L を高速に繰り返すと ON/OFF を繰り返すが，モーターはなめらかに回転する．慣性があるからである．

PWM 信号のデューティ比[10]に応じて，DC モーターの速度が変わる．デューティ比が大きいほど，ON の時間が長くなるので回転速度が速くなる．

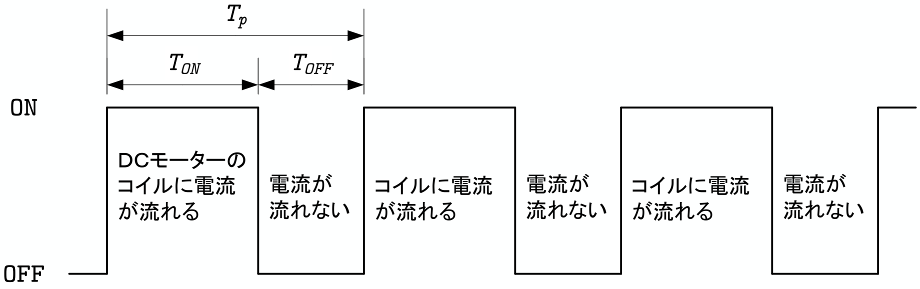


図 16: DC モーターの PWM 制御．DC モーターを高速で ON/OFF を繰り返す．周期 T_p に対する ON 時間 T_{ON} の割合で回転数を変えることができる．

H8 マイコンは，この PWM のための信号出力ができる．ここの実験では，p.10 の図 11 に示しているマイコンの CPU ボードの CN2 の 11 番と 13 番ピンに PWM 信号を出力している．この出力信号をダーリントン接続したトランジスターでスイッチングを行い DC モーターに電力を供給している．出力する PWM 信号のデューティ比はプログラムで決める．

繰り返し周期は，p.27 の B.2.2 節の「制御用の関数」で述べているリスト 4 で決めている．このプログラムでは，繰り返し周期は 32.76 [msec] となる．

A.4 トランジスター

トランジスターは増幅作用を行う半導体素子である．図 17 のように，小さな電流をベースとエミッターの間に流すことにより，それに比例した大きな電流をコレクターとエミッター間に流すことができる．たとえば，マイクの出力を小さい方の電流とし，大きい方をスピーカーに接続する．図 17 の大きい電流の抵抗をスピーカーと考える．すると，マイクからの小さい信号をトランジスターで増幅し，大きな音でスピーカーを動作させることができる．拡声器である．

ここで使っているトランジスターは，バイポーラー型と呼ばれるものである．このタイプのトランジスターの増幅率は，大体数十から数百である．普通に回路を作れば，ベースとエミッター間に 1 [mA] の電流を流すと，コレクターとエミッター間に 100 [mA] 程度の電流を流すことができる．

[10] 周期に対する ON の時間の割合．

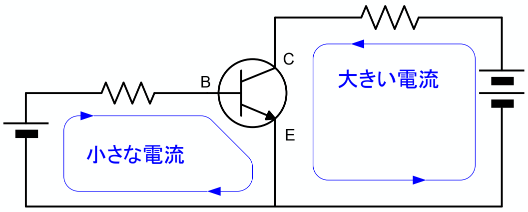


図 17: エミッター接地回路．ベースとエミッターの間に小さな電流を流すことにより，それに比例した大きな電流をコレクターとエミッターの間に流すことができる．

トランジスターはスイッチとして使うことができる．ベースとエミッターの間に流す小さな電流により，スイッチを ON/OFF するように動作させることができる．増幅作用と同じことであるが，小さな電流の方を一定の電流とし，それを流したり/切ったりすることによりあたかもスイッチのように使うのである．その様子を図 18 に示す．

H8 マイコンのように，大きな電流が得られない措置で，モーターのように大電流が必要な機器を ON/OFF するときにトランジスターをスイッチとして使うことができる．H8 マイコンの出力は高々，1 [mA] 程度である．それに対して，ここの実験で使うモーターは数百 [mA] 程度の電流が必要である．直接，H8 マイコンの出力をモーターに接続してもモーターを回転させることはできないし，H8 マイコンが破損するだけである．そこで，H8 マイコンの小電流の出力はトランジスターのスイッチを駆動するためだけ使い，そのスイッチの ON/OFF でモーターを制御する．

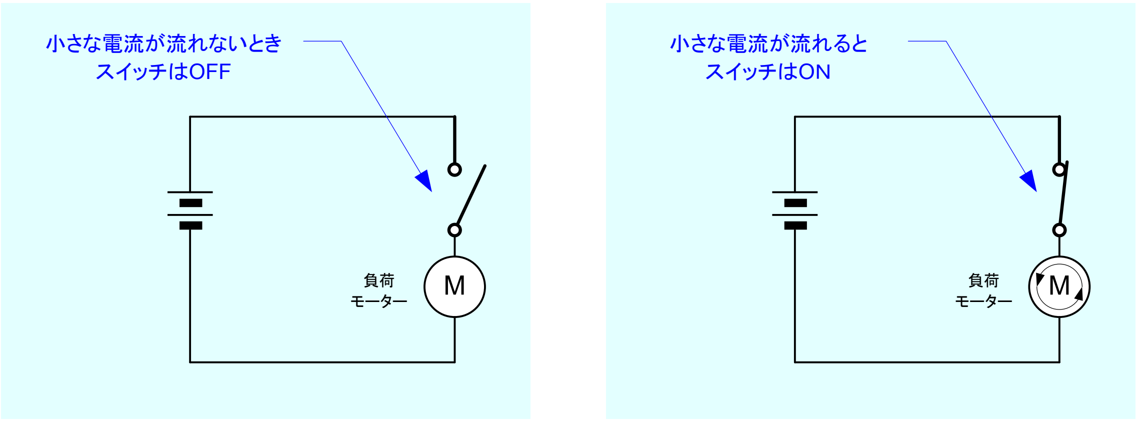


図 18: トランジスターをスイッチとして使って，モーターを制御する様子．H8 マイコンの出力 (小電流) でスイッチを ON/OFF する．

A.5 ダイオード

ダイオードは順方向のみに電流を流す素子である．図 19 のように順方向に接続すると電流が流れ，逆方向だと電流が流れない．順方向だと抵抗がゼロ，逆方向だと抵抗は無限大—と大雑把に考えることができる．

ここの実験では，ダイオードのこの特性を利用して，DC モーターのエネルギー効率の向上に使っている．この様子を図 20 を使って説明する．

- (A) の状態では，ダイオードは逆方向なのでモーターのみに電流が流れる．ダイオードが無い場合と同じように，モーターは回転する．

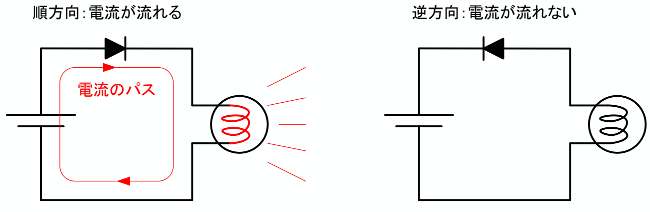


図 19: ダイオードの動作．順方向にダイオードを接続すると電流が流れる．逆方向だと電流が流れない．

- 次にスイッチを OFF にした直後の状態，(B) を考える．このとき，コイルにより作られた磁場のエネルギーがモーター内部に残っている．この磁場のエネルギーがコイルを通して電気のエネルギーに変換される．この電気のエネルギーは，ダイオードを通して流れる電流になる．この電流はモーターに回転力を与える．
- しばらくすると，モーターを回転させるエネルギーが無くなるので，20 の (C) のように停止する．

このようにダイオードを DC モーターに取り付けると，OFF 時の磁場のエネルギーを回転力に変えることができる．したがって，エネルギー効率が上昇するのである．このような働きを行うダイオードを**フリーホイールダイオード**と言う．

さらに，このダイオードにより，逆起電力の緩和の効果もある．スイッチを OFF にした瞬間，コイルの作用により逆起電力が発生する．そうなると，スイッチ—多くの場合半導体—に大きな逆電圧が印可される．ダイオードがあると，図 20 の (B) のように電流が流れ，スイッチには大きな逆電圧が印可されない．なぜならば，ダイオードの抵抗はほぼゼロと考えることができるからである．

スイッチの ON/OFF の回数が多い場合に，フリーホイールダイオードの効果が現れる．ここの実験では PWM 制御で，1 秒間に 30 回，ON/OFF を繰り返している．十分効果が期待できる．DC モーターを PWM 制御する場合，必ず取り付けるべきである．

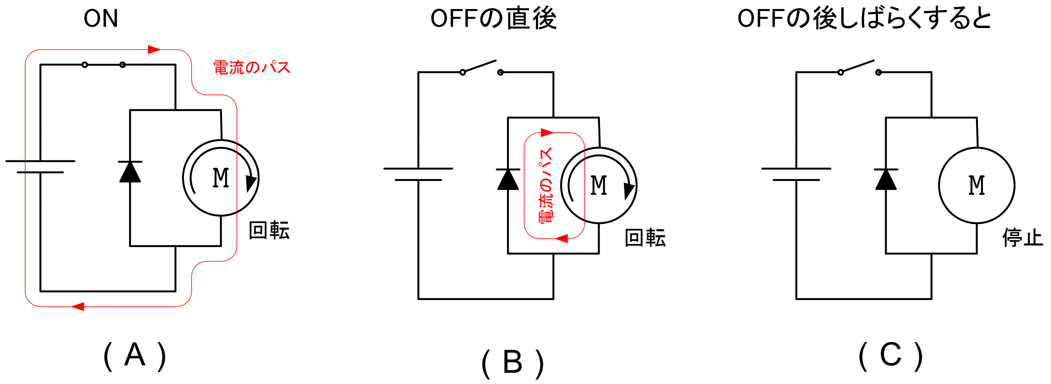


図 20: 電源とスイッチ，ダイオード，直流モーターから成る回路である．スイッチが ON の時，電流はダイオードに流れず，モーターのみに流れる．モータは回転する．スイッチを OFF にした瞬間，モーターのコイルの磁場のエネルギーにより電流がダイオードを通して流れ，モーターの回転のエネルギーに変換される．しばらくすると，回転に寄与するエネルギーが無くなりモーターは停止する．

B　ライントレースカーのソフトウェアー

B.1　メイクファイル

プログラムの実行ファイルは複数のソースファイルから作られることが多い．このようなとき，コンパイル作業はかなり面倒である．この面倒な作業を自動的に行わせるために，`make` がある．ここで使うプログラムも，C 言語，アセンブラ言語，各種ヘッダーファイル，スタートアップルーチンのファイルから構成される．そのため，`make` を使い，自動的にコンパイルできるようにしている．リスト 2 の Makefile は，コンパイルのこのコンパイルの方法を記述している．

また，H8 マイコンへのバイナリーデータ転送の面倒な処理も `make` でできる．このようにすることにより，諸君の作業は格段に楽になる．

この実験で使う Makefile には，次の機能がある．

表 5: Makefile の機能

コマンド	実行内容
`make`	ソースファイルをコンパイルリンクして，転送用の mot ファイルを作成する．
`make clean`	オブジェクトファイルと `exe`，mot ファイルを削除する．
`make write`	mot ファイルを H8 マイコンへ転送する．

リスト 2: ライントレースカー用の Makefile(Makefile)

```
TARGET          =       h8exp.mot

CFLAGS          =       -O -mh -g -mrelax -mint32 -DH8_3664
TOOL_PREFIX     =       h8300-hms-
CC              =       $(TOOL_PREFIX)gcc

LDSCRIPT        =       h8_link.x
CRT0            =       h8_start.s
SRCS            =       experiment.c
FUNC            =       h8c.c

all : $(TARGET)

$(TARGET): $(TARGET:.mot=.exe)
        $(TOOL_PREFIX)objcopy -O srec $(TARGET:.mot=.exe) $@

$(TARGET:.mot=.exe): Makefile $(LDSCRIPT) $(SRCS:.c=.o) $(FUNC:.c=.o) \
$(CRT0) $(LIBS)
        $(CC) $(CFLAGS) -T $(LDSCRIPT) -nostdlib $(CRT0) \
$(SRCS:.c=.o) $(FUNC:.c=.o) -o $@ $(LIBS) -lc -lgcc

write:
        h8write -3664 $(TARGET) /dev/ttyS0

clean :
        rm -f $(TARGET)
        rm -f $(TARGET:.mot=.exe)
        rm -f $(SRCS:.c=.o)
        rm -f $(FUNC:.c=.o)
```

B.2 ライントレースカー用の関数

ライントレースカーの制御の変更は，p.12 に示したリスト 1 を変えることによりできる．しかし，ライントレーサーを制御を司る H8 マイコンのプログラムは，リスト 1 のみでできているわけではない．これから示すさまざまなプログラムからできあがっている．

B.2.1 ヘッダーファイル

ここのライントレースカーを制御するために 6 個の専用の関数をリスト 4 で作成している．リスト 3 のヘッダーファイルでは，その関数のプロトタイプ宣言やグローバル変数を定義している．

リスト 3: ライントレースカー用関数のヘッダーファイル (h8c.h)

```
#ifndef H8C_H
#define H8C_H

#define _nop() asm volatile ("nop");

void init_led(void);
void init_pwm(void);
void start_pwm(void);
void duty_pwm(unsigned int b, unsigned int d);
void ADC_init(void);
void ADC(void);

typedef union{
  unsigned char BYTE;
  struct{
    unsigned char ADF:1;
    unsigned char ADIE:1;
    unsigned char ADST:1;
    unsigned char SCAN:1;
    unsigned char CKS:1;
    unsigned char CH2:1;
    unsigned char CH1:1;
    unsigned char CH0:1;
  }BIT;
}set_ADC;

int ch1, ch2, ch3, ch4;

#endif
```

B.2.2 制御用の関数

リスト 3 のヘッダーファイルでプロトタイプ宣言した関数の実装をリスト 4 のプログラムで行っている．マイコンを実行させるためには多くのビット操作が必要なので，このように別ファイルの別関数にするとその操作を隠すことができる．ユーザーにはその関数の実行方法のみを示すことにより，容易にプログラムができるようになる．

リスト 4: ライントレースカー用関数のファイル (h8c.c)

```
#include "3664.h"
#include "h8c.h"


//=============================================================
// LED 点灯のための初期化関数
//     Port 5 を使うときに最初，回だけに呼び出す1
//     PDR 5 のビットの設定で，点灯するを決めるLED
//=============================================================
```

```
void init_led(void){
  PMR5 &= 0xc0;          // Port Mode Register 5 の設定
                         //     0 : 汎用入出力 1 : マニュアルを見よ
  PCR5 |= 0x3f;          // Port Control Register 5 の設定
                         //     0 : 入力 1 : 出力 */
  PDR5 &= 0xc0;          // Port Data Register
}


//============================================================
// PWM を使うための初期化関数
//     PWM を使うときに，最初回呼び出す1
//============================================================
void init_pwm(void){
  TMRW |= 0x05;          /* FTIOB と FTIOD 端子を PWM 出力に */

  TCRW |= 0x80;          /* コンペアマッチ A によりカウンタクリアー */
  TCRW |= 0x30;          /* クロックセレクト内部クロック  phi /8 (=2MHz)*/
  TCRW |= 0x0a;          /* タイマ出力セットレベルコンペアマッチまで 1 */

  TCNT = 0x0000;         /* カウンタビットの初期値16 */

  GRA = 0xffff;

}


//============================================================
// PWM の Duty を決める
// 第一引数   : FTIOB 第二引数  : FTIOD
//============================================================
void duty_pwm(unsigned int b, unsigned int d){   // pwm デューティ [%]

  GRB = (unsigned int)655*b;
  GRD = (unsigned int)655*d;

}


//============================================================
// PWM のスタート時に呼び出す
//============================================================
void start_pwm(void){
  TMRW |= 0x80;          // カウンタースタート
}


//============================================================
// AD 変換器の初期化関数
//============================================================
void init_ADC(void)
{

  set_ADC *set;

  set=(set_ADC *)&ADCSR;

  set->BIT.ADF=0;
  set->BIT.ADIE=0;
  set->BIT.ADST=0;
  set->BIT.SCAN=1;
  set->BIT.CKS=0;
  set->BIT.CH2=0;
  set->BIT.CH1=1;
  set->BIT.CH0=1;

}

//============================================================
// AD 変換器を行い結果を設定
//       ch1:AN0 の結果  0 : 白 1 : 黒
```

```
//      ch2:AN1 の結果  0 : 白 1 : 黒
//      ch3:AN2 の結果  0 : 白 1 : 黒
//      ch4:AN3 の結果  0 : 白 1 : 黒
//============================================================
void ADC(void)
{

  ADCSR &=0x7f;              // ADF=0   AD 完了フラグをゼロに
  ADCSR |= 0x20;             // ADST=1  ADC スタート

  while(!(ADCSR & 0x80));    // AD 変換完了まで待つ
  ADCSR &= 0xdf;             // ADST=0  ADC stop

  ch1=(unsigned int)(ADDRA >> 6)<200 ? 1:0;
  ch2=(unsigned int)(ADDRB >> 6)<200 ? 1:0;
  ch3=(unsigned int)(ADDRC >> 6)<200 ? 1:0;
  ch4=(unsigned int)(ADDRD >> 6)<200 ? 1:0;

}
```

B.3 H8 マイコン 3664 のヘッダーファイル

リスト 5 のヘッダーファイル (3664.h) は，一部のアセンブラ言語で書かれた関数とレジスターのアドレスを定義している．

3–5 行目がアセンブラで命令を記述している．マクロ名を指定することにで，アセンブラの命令が実行される．簡単なアセンブラの命令を実行するときに便利である．

14–96 行目でレジスターのアドレスを定義している．レジスター名—ここではマクロ名になるが—とそれに対応するアドレスは，H8 のマニュアルに書いてある．このようにすると，無味乾燥な 16 進数のアドレスが意味のある名前に置き換えることができる．プログラムが書きやすくなる．コンパイル時に gcc のプリプロセッサーがこのマクロ名を対応する 16 進数に変換する．

レジスターとアドレスの関係は，ルネサステクノロジー社の「H8/3664 グループハードウェアーマニュアル」[3] に詳しく書かれている．

リスト 5: H8/3664N 用のヘッダーファイル (3664.h)．アセンブラで記述した関数 (マクロ) とレジスター名を決めている．

```
1  #define STI()    asm volatile ("andc.b #0x7f,ccr")
2  #define CLI()    asm volatile ("orc.b #0x80,ccr")
3  #define SLEEP()  asm volatile ("sleep")
4
5  #define VOLATILE_BYTE(addr)   (*(volatile unsigned char*)(addr))
6  #define VOLATILE_WORD(addr)   (*(volatile unsigned short*)(addr))
7
8  //=====================================================================
9  //      Definitions of H8/3664 Internal Registers
10 //      DO NOT EDIT!
11 //=====================================================================
12 #define TMRW    VOLATILE_BYTE(0xff80)
13 #define TCRW    VOLATILE_BYTE(0xff81)
14 #define TIERW   VOLATILE_BYTE(0xff82)
15 #define TSRW    VOLATILE_BYTE(0xff83)
16 #define TIOR0   VOLATILE_BYTE(0xff84)
17 #define TIOR1   VOLATILE_BYTE(0xff85)
18
19 #define TCNT    VOLATILE_WORD(0xff86)
20 #define GRA     VOLATILE_WORD(0xff88)
21 #define GRB     VOLATILE_WORD(0xff8a)
22 #define GRC     VOLATILE_WORD(0xff8c)
23 #define GRD     VOLATILE_WORD(0xff8e)
24
25 #define FLMCR1  VOLATILE_BYTE(0xff90)
26 #define FLMCR2  VOLATILE_BYTE(0xff91)
```

```c
#define FLPWCR  VOLATILE_BYTE(0xff92)
#define EBR1    VOLATILE_BYTE(0xff93)
#define FENR    VOLATILE_BYTE(0xff9b)
#define TCRV0   VOLATILE_BYTE(0xffa0)
#define TCSRV   VOLATILE_BYTE(0xffa1)
#define TCORA   VOLATILE_BYTE(0xffa2)
#define TCORB   VOLATILE_BYTE(0xffa3)
#define TCNTV   VOLATILE_BYTE(0xffa4)
#define TCRV1   VOLATILE_BYTE(0xffa5)
#define TMA     VOLATILE_BYTE(0xffa6)
#define TCA     VOLATILE_BYTE(0xffa7)
#define SMR     VOLATILE_BYTE(0xffa8)
#define BRR     VOLATILE_BYTE(0xffa9)
#define SCR3    VOLATILE_BYTE(0xffaa)
#define TDR     VOLATILE_BYTE(0xffab)

#define SSR     VOLATILE_BYTE(0xffac)
#define RDR     VOLATILE_BYTE(0xffad)

#define ADDRA   VOLATILE_WORD(0xffb0)
#define ADDRB   VOLATILE_WORD(0xffb2)
#define ADDRC   VOLATILE_WORD(0xffb4)
#define ADDRD   VOLATILE_WORD(0xffb6)
#define ADCSR   VOLATILE_BYTE(0xffb8)
#define ADCR    VOLATILE_BYTE(0xffb9)

#define TCSRWD  VOLATILE_BYTE(0xffc0)
#define TCWD    VOLATILE_BYTE(0xffc1)
#define TMWD    VOLATILE_BYTE(0xffc2)
#define ICCR    VOLATILE_BYTE(0xffc4)
#define ICSR    VOLATILE_BYTE(0xffc5)
#define ICDR    VOLATILE_BYTE(0xffc6)
#define SARX    VOLATILE_BYTE(0xffc6)
#define ICMR    VOLATILE_BYTE(0xffc7)
#define SAR     VOLATILE_BYTE(0xffc7)
#define ABRKCR  VOLATILE_BYTE(0xffc8)
#define ABRKSR  VOLATILE_BYTE(0xffc9)
#define BARH    VOLATILE_BYTE(0xffca)
#define BARL    VOLATILE_BYTE(0xffcb)
#define BDRH    VOLATILE_BYTE(0xffcc)
#define BDRL    VOLATILE_BYTE(0xffcd)
#define PUCR1   VOLATILE_BYTE(0xffd0)
#define PUCR5   VOLATILE_BYTE(0xffd1)
#define PDR1    VOLATILE_BYTE(0xffd4)
#define PDR2    VOLATILE_BYTE(0xffd5)
#define PDR5    VOLATILE_BYTE(0xffd8)
#define PDR7    VOLATILE_BYTE(0xffda)
#define PDR8    VOLATILE_BYTE(0xffdb)
#define PDRB    VOLATILE_BYTE(0xffdd)

#define PMR1    VOLATILE_BYTE(0xffe0)
#define PMR5    VOLATILE_BYTE(0xffe1)
#define PCR1    VOLATILE_BYTE(0xffe4)
#define PCR2    VOLATILE_BYTE(0xffe5)
#define PCR5    VOLATILE_BYTE(0xffe8)
#define PCR7    VOLATILE_BYTE(0xffea)
#define PCR8    VOLATILE_BYTE(0xffeb)
#define SYSCR1  VOLATILE_BYTE(0xfff0)
#define SYSCR2  VOLATILE_BYTE(0xfff1)
#define IEGR1   VOLATILE_BYTE(0xfff2)
#define IEGR2   VOLATILE_BYTE(0xfff3)
#define IENR1   VOLATILE_BYTE(0xfff4)
#define IRR1    VOLATILE_BYTE(0xfff6)
#define IWPR    VOLATILE_BYTE(0xfff8)
#define MSTCR1  VOLATILE_BYTE(0xfff9)
#define TSCR    VOLATILE_BYTE(0xfffc)

#define EKR     VOLATILE_BYTE(0xff10)
```

B.4 スタートアップルーチン

H8 マイコンに電源を入れたときやリセットスイッチを押したとき，実行されるプログラムをスタートアップルーチンと言う．リスト 6 が，ここで使っているスタートアップルーチンである．

このスタートアップルーチンでは，まずはスタックポインターを設定している．リンカースクリプトで設定されたスタック領域の先頭アドレス (_stack) をスタックポインター (sp) に設定している．次に ROM 領域のデータを RAM 領域に転送している．このようにすることにより，初期値がある変数に対応している．その後，リスト 1 を main 関数に実行を移している．実際にはこのメイン関数は機械語に翻訳され，ラベルは_main となる．ここでは，jsr でジャンプしている．この_main から戻るようなことがあると，無限ループに入るようにしている．

リスト 6: スタートアップルーチン (h8_start.s).

```
        ;; H8/3664 Startup routine

        .h8300h
        .section .text
        .global _start
_start:
        mov.l #_stack,sp   ;; スタックポインターの設定

        ;; ROM 領域のデータを RAM へ転送

        mov.l #___dtors_end,er0  ;; .dtors の終わりのアドレス
        mov.l #___data,er1  ;; .data の開始アドレス
        mov.l #_edata,er2  ;; .data の終わりのアドレス
ram_data:
        mov.w @er0,r3
        mov.w r3,@er1
        adds #2,er0
        adds #2,er1
        cmp.l er2,er1
        blo ram_data

        ;; main 関数の実行

        jsr @_main
sleep:
        sleep
        bra sleep

        .end
```

B.5 リンカースクリプト

リンカースクリプトは，プログラムのメモリーの配置の仕方を決めるものである．

リスト 7: リンカースクリプト (h8_link.x).

```
OUTPUT_FORMAT("coff-h8300")
OUTPUT_ARCH(h8300h)
ENTRY("_start")
MEMORY
{
        vectors  : o = 0x0000, l = 0x0034
        rom      : o = 0x0034, l = 0x7fcc
        ram      : o = 0xf780, l = 0x0400
        stack    : o = 0xff80, l = 0x0000
}

SECTIONS
{
.vectors : {
        SHORT(ABSOLUTE(_start))
```

```
        SHORT(ABSOLUTE(_start))
        SHORT(ABSOLUTE(_start))
        SHORT(ABSOLUTE(_start))
        SHORT(ABSOLUTE(_start))
        SHORT(ABSOLUTE(_start))
        SHORT(ABSOLUTE(_start))
        SHORT(DEFINED(_int_nmi)?ABSOLUTE(_int_nmi):ABSOLUTE(_start))
        SHORT(DEFINED(_int_trap0)?ABSOLUTE(_int_trap0):ABSOLUTE(_start))
        SHORT(DEFINED(_int_trap1)?ABSOLUTE(_int_trap1):ABSOLUTE(_start))
        SHORT(DEFINED(_int_trap2)?ABSOLUTE(_int_trap2):ABSOLUTE(_start))
        SHORT(DEFINED(_int_trap3)?ABSOLUTE(_int_trap3):ABSOLUTE(_start))
        SHORT(DEFINED(_int_break)?ABSOLUTE(_int_break):ABSOLUTE(_start))
        SHORT(DEFINED(_int_sleep)?ABSOLUTE(_int_sleep):ABSOLUTE(_start))
        SHORT(DEFINED(_int_irq0)?ABSOLUTE(_int_irq0):ABSOLUTE(_start))
        SHORT(DEFINED(_int_irq1)?ABSOLUTE(_int_irq1):ABSOLUTE(_start))
        SHORT(DEFINED(_int_irq2)?ABSOLUTE(_int_irq2):ABSOLUTE(_start))
        SHORT(DEFINED(_int_irq3)?ABSOLUTE(_int_irq3):ABSOLUTE(_start))
        SHORT(DEFINED(_int_wkp)?ABSOLUTE(_int_wkp):ABSOLUTE(_start))
        SHORT(DEFINED(_int_timera)?ABSOLUTE(_int_timera):ABSOLUTE(_start))
        SHORT(ABSOLUTE(_start))
        SHORT(DEFINED(_int_timerw)?ABSOLUTE(_int_timerw):ABSOLUTE(_start))
        SHORT(DEFINED(_int_timerv)?ABSOLUTE(_int_timerv):ABSOLUTE(_start))
        SHORT(DEFINED(_int_sci3)?ABSOLUTE(_int_sci3):ABSOLUTE(_start))
        SHORT(DEFINED(_int_iic)?ABSOLUTE(_int_iic):ABSOLUTE(_start))
        SHORT(DEFINED(_int_adi)?ABSOLUTE(_int_adi):ABSOLUTE(_start))
        }  > vectors

.text 0x0034 : {
        *(.text)
        *(.strings)
        *(.rodata)
         _etext = . ;
        } > rom

.tors : {
        ___ctors = . ;
        *(.ctors)
        ___ctors_end = . ;
        ___dtors = . ;
        *(.dtors)
        ___dtors_end = . ;
        }  > rom

.data : AT ( ADDR(.tors) + SIZEOF(.tors) ){
        ___data = . ;
        *(.data)
        *(.tiny)
         _edata = .;
        } > rom

.bss : AT ( LOADADDR(.data) + SIZEOF(.data) ) {
         _bss_start = . ;
        *(.bss)
        *(COMMON)
         _end = . ;
        }  >ram

.stack : {
         _stack = . ;
        *(.stack)
        }  > stack

.stab 0 (NOLOAD) : {
        [ .stab ]
        }

.stabstr 0 (NOLOAD) : {
        [ .stabstr ]
        }
}
```

C　H8 マイコンキットの作り方

実験でつかう H8 マイコンキットの部品を表 6 に示す．H8 マイコンキットの電圧は，前期の実験では 5 V，後期では 9 V をつかう．このように 2 種類の電源を使うため，マイコンキットの配線パターンを一部を切り離す必要がある．これらの方法を示す．また，実験に使う H8 マイコンキットは，ブレッドボードに取り付けるために，特別なピン (電極) を使う必要がある．

表 6: 実験に使う H8 マイコンキットとその部品

名称	業者	型番	備考
AKI-H8/366N タイニーマイコンキット	秋月電子	HD64N3664FP	
ピンヘッダ (オス)40P(1×40)	秋月電子		通販コード C-167
丸ピン IC ソケット (シングル 40P)	秋月電子	6604S-40	通販コード P-1591

JP1 のパターンの切り離し

部品表に示した秋月電子の H8 マイコンキットの通常の入力電圧は 7–12 V である．これを，3 端子レギュレーターで 5 V に落として，マイコン本体に供給している．このマイコンキットの取扱説明書に書かれている通り，JP1 のパターンを切る．すると，つぎのような使い方ができる．

- JP1 のジャンパーピンを OFF(オープン)．CN1 の 23 番ピンに 5 V，25 と 26 番ピンをグランド (0 V) で使うことができる．
- JP1 のジャンパーピンを ON(クローズ)．CN1 の 24 番ピンに 7–12 V，25 と 26 番ピンをグランド (0 V) で使うことができる．

ピンの取り付け

CN1 と CN2 の外側にピンヘッダ (オス)，内側に丸ピン IC ソケットを取り付ける．

参考文献

[1] 鹿取佑二．C 言語で H8 マイコンをつかいこなす.（株）オーム社，2005.

[2] 木下是雄．”理科系の作文技術．”中公新書．中央公論社，1994.

[3] H8/3664 グループハードウェアーマニュアル（rjj09b0149-066）